CLAIM AMENDMENTS:

1. (Previously presented) A method for performing memory disambiguation in a compiler, comprising:

determining memory objects corresponding to memory references in one or more source files being compiled;

creating a unique memory disambiguation token for each of the memory references, each unique memory disambiguation token comprising a data structure including a plurality of links to data objects in which disambiguation information are stored, each unique memory disambiguation token identifying information particular to the memory reference it is associated with so as to preserve high-level and intermediate-level semantic information;

creating a symbolic memory reference representation associated with each unique memory disambiguation token, including information on whether the memory reference is indirect or direct and access to symbol table information for a pointer to the memory object for indirect references or the memory object for direct references; and

determining if potentially dependent memory references are dependent or independent based on information contained in the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

2. (Previously presented) The method of claim 1, further comprising determining if the memory references are redundant based on information contained in the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

3. (Original) The method of claim 1, further comprising determining a relative difference in starting addresses for two memory references that are determined to be independent or dependent.

4. (Canceled) Please cancel Claim 4 without prejudice.

5. (Previously presented) The method of claim 1, wherein the data structure is embedded in memory reference operators of an intermediate language produced during the compilation of the one or more source files.

6. (Previously presented) The method of claim 1, wherein the unique memory disambiguation token associated with the memory object includes a key that is used to access a table of data dependence information.

7. (Previously presented) The method of claim 1, wherein the unique memory disambiguation token contains a link to address base and offset information for the memory reference that is used for low-level disambiguation.

8. (Original) The method of claim 1, further comprising:

substituting a direct memory reference for an indirect memory reference; and

updating the disambiguation token corresponding to the memory reference to indicate the memory reference is now a direct memory reference.

9. (Original) The method of claim 1, further comprising using information identified by disambiguation tokens to determine sets of local memory objects that are not referenced after they are modified.

10. (Original) The method of claim 1, further comprising determining if two memory references access overlapping memory locations based on information contained in the disambiguation tokens for those memory references and their associated symbolic memory reference representations.

11. (Original) The method of claim 10, further comprising determining particularities of an overlap between two overlapping memory references.

12. (Previously presented) The method of claim 1, further comprising:

determining functions executed corresponding to function calls in the one or more source files being compiled;

creating a disambiguation token for each function call, each disambiguation token identifying information particular to the function call it is associated with so as to preserve high-level and intermediate level semantic information;

creating a symbolic function call representation associated with each disambiguation token, including information on whether the function call is indirect or direct and access to symbol table information for the pointer or function respectively; and

determining if potentially dependent calls and memory references are dependent or independent for the function calls based on information contained in the disambiguation tokens for the calls and memory references, their associated symbolic representation, an analysis of each function to determine the set of memory locations modified or referenced by the function.

13. (Original) The method of claim 1, wherein the disambiguation token contains a link to type information associated with the memory reference.

14. (Original) The method of claim 1, wherein the disambiguation token for an indirect memory reference contains a link to a set of memory objects accessible via the pointer as determined by points-to analysis.

15. (Previously presented) The method of claim 1, further comprising using the unique memory disambiguation token and the symbolic memory reference representation as an interface or means of communication between various software components of a disambiguator that performs memory disambiguation functions and clients of the disambiguator.

16. (Previously presented) A system comprising:

a memory in which a plurality of machine instructions comprising a compiler and programming code corresponding to one or more source files are stored; and

a processor coupled to the memory, executing machine instructions to perform the functions of:

determining memory objects corresponding to memory references in said one or more source files;

creating a unique memory disambiguation token for each of the memory references, each unique memory disambiguation token comprising a data structure including a plurality of links to data objects in which disambiguation information are stored, each unique memory disambiguation token identifying information particular to the memory reference it is associated with so as to preserve high-level and intermediate-level semantic information

creating a symbolic memory reference representation associated with each unique memory disambiguation token, including information on whether the memory reference is indirect or direct and access to symbol table information for a pointer to the memory object for indirect references or the memory object for direct references; and

determining if potentially dependent memory references are dependent or independent based on information contained in the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

17. (Previously presented) The system of claim 16, wherein execution of the machine instructions by the processor further performs the function of determining if the memory references are redundant based on information contained in the disambiguation tokens for the memory references and their associated symbolic memory reference representations.

18. (Original) The system of claim 16, wherein execution of the machine instructions by the processor further performs the function of determining relative positions of starting addresses for memory references that are independent or dependent.

19. (Previously presented) The system of claim 16, wherein execution of the machine instructions by the processor further performs the functions of:

substituting a direct memory reference for an indirect memory reference; and

updating the unique memory disambiguation token corresponding to the memory reference to indicate the memory reference is now a direct memory reference.

20. (Previously presented) The system of claim 16, wherein execution of the machine instructions by the processor further performs the function of using information identified by each of the unique memory disambiguation tokens to determine sets of local memory objects that are not referenced after they are modified.

21. (Previously presented) The system of claim 16, wherein execution of the machine instructions by the processor further performs the function of determining if two memory references access overlapping memory locations based on information contained in each of the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

22. (Original) The system of claim 16, wherein execution of the machine instructions by the processor further performs the functions of:

determining the functions executed corresponding to function calls in the one or more source files being compiled;

creating a disambiguation token for each of the function calls, each disambiguation token identifying information particular to the function call it is associated with so as to preserve high-level and intermediate level semantic information;

creating a symbolic function call representation associated with each disambiguation token, including information on whether the function call is indirect or direct and access to symbol table information for the pointer or function respectively; and

determining if potentially dependent calls and memory references are dependent or independent for the function calls based on information contained in the disambiguation tokens for the calls and memory references, their associated symbolic representation, an analysis of each function to determine the set of memory locations modified or referenced by the function.

23. (Previously presented) An article of manufacture on which a plurality of machine instructions comprising a compiler are stored that upon execution of the machine instructions by a processor causes the functions to be performed, including:

determining memory objects corresponding to memory references in said one or more source files;

creating a unique memory disambiguation token for each of the memory references, each unique memory disambiguation token comprising a data structure including a plurality of links to data objects in which disambiguation information are stored, each unique memory disambiguation token identifying information particular to the memory reference it is associated with so as to preserve high-level and intermediate-level semantic information

creating a symbolic memory reference representation associated with each unique memory disambiguation token, including information on whether the memory reference is indirect or direct and access to symbol table information for a pointer to the memory object for indirect references or the memory object for direct references; and

determining if potentially dependent memory references are dependent or independent based on information contained in the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

24. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the function of determining if the memory references are redundant based on information contained in the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

25. (Original) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the function of determining relative positions of starting addresses for memory references that are independent or dependent.

26. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the functions of:

substituting a direct memory reference for an indirect memory reference; and

updating the unique memory disambiguation token corresponding to the memory reference to indicate the memory reference is now a direct memory reference.

27. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the function of using information identified by each of the unique memory disambiguation tokens to determine sets of local memory objects that are not referenced after they are modified.

28. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the function of determining if two memory references access overlapping memory locations based on information contained in each of the unique memory disambiguation tokens for those memory references and their associated symbolic memory reference representations.

29. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the functions of:

determining the functions executed corresponding to function calls in the one or more source files being compiled;

creating a disambiguation token for each of the function calls, each disambiguation token identifying information particular to the function call it is associated with so as to preserve high-level and intermediate level semantic information;

creating a symbolic function call representation associated with each disambiguation token, including information on whether the function call is indirect or direct and access to symbol table information for the pointer or function respectively; and

determining if potentially dependent calls and memory references are dependent or independent for the function calls based on information contained in the disambiguation tokens for the calls and memory references, their associated symbolic

representation, an analysis of each function to determine the set of memory locations modified or referenced by the function.

30. (Previously presented) The article of manufacture of claim 23, wherein execution of the machine instructions further performs the functions of using the unique memory disambiguation token and the symbolic memory reference representation as an interface or means of communication between various software components of a disambiguator that performs memory disambiguation functions and clients of the disambiguator.